RELIABILITY INVESTIGATIONS OF HETEROGENEOUS TERMINAL SYSTEMS USING MOSEL*

B. Almási and J. Sztrik (Debrecen, Hungary) UDC 519.2

1. Introduction

Performance modeling of computer and communication systems with the help of different stochastic methods and queueing theory has been widely accepted in computer engineering (see, for example, [5, 6, 14]). Several works have been devoted to the investigation of the utilization factor of the Central Processor Unit (CPU) and the number of jobs staying at the CPU. It has turned out that in the case where the involved random variables are exponentially distributed, the request generation rates are the same, and the processing rates are different, Lehtonen in [12] and Van der Wal in [15] have proved that the utilization of the CPU is not influenced at all by any work-conserving scheduling rule, including First In First Out (FIFO), Processor Sharing (PS), Priority Processor Sharing (PPS), and Preemptive or Nonpreemptive priority disciplines. More precisely, it has been shown that the mean busy-period length of the processor is the same for any of the above-mentioned scheduling. Furthermore, the mean number of jobs staying at the CPU is maximized by giving higher preemptive priority to a job with less mean job size (so-called H-schedule). Consequently, the overall utilization of the system, the sum of CPU and terminal utilisations, sometimes called the effective degree of multiprogramming, is maximized. In the case where the request generation rates are also different, by using different methods, Koole in [9] and Van der Wal in [15] have shown that if preemptions of the resume type are allowed, the CPU utilization is maximized by giving higher priority to the jobs of the faster-thinking terminals irrespectively of the expected job sizes.

However, in practice, we can see that the terminals and the CPU are not always available for service. These situations could be considered as breakdowns, so the reliability analysis of nonreliable terminal systems seems to be important also. To do this, we can use a wide range of mathematical methods accumulated in recent years (cf. [2, 4, 8, 10, 11, 13]).

Thus, the aim of the present paper is to investigate the influence of breakdowns of the CPU and terminals on the main performance measures of the system. Using MOSEL, the problem can be formulated and solved automatically and different graphics can demonstrate the above-mentioned effects.

2. Modeling Nonreliable, Nonhomogeneous Terminal Systems

2.1. The mathematical model. Let us consider a terminal system consisting of \( n \) terminals connected with a Central Processing Unit (CPU) where three service disciplines can be employed: FIFO, Priority Processor Sharing (PPS), and Polling. At the terminals there is no queuing delay for the jobs (as the number of terminals is equal to the number of jobs) and a user at terminal \( i \) has thinking (i.e., program generation) times and processing (i.e., program running) times depending on the index \( i \). We assume that each user generates only one job at a time, and he waits until the CPU services it. It is assumed that the busy terminals (that is, when the user is generating a new job) and the CPU (either it is busy or idle) are subject to random breakdowns, thus giving duties to a single repairman. The random working and repair times of the terminals are exponentially distributed with mean depending on the terminal index (nonhomogeneous breakdowns). Furthermore, we assume that the CPU is responsible for the system work, i.e., service stops at the terminals and at the CPU when the CPU is down. The repairman gives preemptive priority to the CPU failure and follows the FIFO discipline for the terminal breakdowns.

Let us denote by \( \lambda_i, \mu_i, \gamma_i, \tau_i, \) and \( \omega_i \) the parameters of the exponentially distributed thinking, processing, operating, repair times, and weight for terminal \( i, i = 1, \ldots, n \), respectively. Similarly, let \( \alpha \) and \( \beta \) denote the failure and repair rate of the CPU, respectively. The random variables are assumed to be independent of each other.

---

*Partially supported by the Hungarian Scientific-Research Fund (grant No. T034280/2000) and German–Hungarian Bilateral Intergovernmental Scientific Cooperation (grant No. 21-2000), and FKFP (grant No. 0191/2001).

To deal with the problem, we have to introduce the following random variables: 

- \( X(t) \) is 1 if the operating system fails at time \( t \), or 0 otherwise;
- \( Y(t) \) is the failed terminals' indices at time \( t \) in order of their failure, or 0 if there is no failed terminal;
- \( Z(t) \) is the indices of the jobs residing at the CPU at time \( t \), or 0 if the CPU is idle.

Depending on the service discipline, the random variable \( Z(t) \) gives the order of service by the CPU, too. It can easily be seen that the stochastic process \( M(t) = (X(t), Y(t), Z(t)) \) is a Markov chain having a rather complex and large state space. To get its steady-state probabilities, an efficient recursive computational method has been introduced and used for the different service rules mentioned earlier (cf. [1]). In the present case, MOSEL (Modeling, Specification and Evaluation Language), developed at the University of Erlangen, Germany, is used to formulate and solve the problem (see [3]). It automatically generates the state probabilities and, using them, a result file containing the performance measures specified in the model description file. It also generates a graphical representation of the results if the input file contains a picture. Since in our paper we concentrate on the effect of different parameters of the system on the main performance measures, the technical details of programming are not treated.

Let us denote the steady-state distribution of \( (M(t), t > 0) \) by

\[
p(q; i_1, \ldots, i_k; j_1, \ldots, j_s) = \lim_{t \to \infty} p(X(t) = q; Y(t) = i_1, \ldots, i_k; Z(t) = j_1, \ldots, j_s).
\]

Furthermore, let us denote by \( p(q, k, s) \) the steady-state probability that the operating system is in state \( q \), \( k \) terminals have failed, and \( s \) jobs are at the CPU.

Knowing these probabilities, the main performance measures can be obtained as follows:

(i) Mean number of jobs residing at the CPU

\[
\bar{\pi}_j = \frac{1}{n} \sum_{i=0}^{n-k} \sum_{k=0}^{n-k} \sum_{s=0}^{n-k} sp(i, k, s).
\]

(ii) Mean number of good terminals

\[
\bar{\pi}_g = n - \frac{1}{n} \sum_{i=0}^{n-k} \sum_{k=0}^{n-k} \sum_{s=0}^{n-k} kp(i, k, s).
\]

(iii) Average number of busy terminals

\[
\bar{\pi}_b = \frac{n}{n-k} \sum_{k=0}^{n-k} \sum_{s=0}^{n-k} (n - k - s)p(0, k, s).
\]

(iv) Utilization of the repairman

\[
U_r = \frac{n}{n-k} \sum_{k=0}^{n-k} \sum_{s=0}^{n-k} p(1, k, s) + \sum_{k=1}^{n-k} \sum_{s=0}^{n-k} p(0, k, s).
\]

(v) Utilization of the CPU

\[
U_{CPU} = \frac{n}{n-k} \sum_{k=0}^{n-k} \sum_{s=1}^{n-k} p(0, k, s).
\]

(vi) Utilization of terminal \( i, i = 1, \ldots, n, \)

\[
U_i = \frac{n}{n-k} \sum_{k=0}^{n-k} \sum_{s=0}^{n-k} \sum_{i_1, \ldots, i_k} \sum_{j_1, \ldots, j_k} \left( \prod_{r=1}^{k} \prod_{v=1}^{s} (1 - \delta(i, i_r) - \delta(i, j_v)) \right) p(0; i_1, \ldots, i_k; j_1, \ldots, j_s).
\]

(vii) Expected response time of jobs for terminal \( i \)

\[
T_i = \frac{Q_i}{\lambda_i U_i},
\]

where

\[
\delta(i, j) = \begin{cases} 
1, & \text{if } i = j, \\
0, & \text{otherwise},
\end{cases}
\]

and \( Q_i \) denotes the probability of staying at the CPU for terminal \( i \), namely,
Fig. 1. Utilizations versus CPU breakdown intensity in the FIFO case.

Fig. 2. Mean response times versus CPU breakdown intensity in the FIFO case.

Fig. 3. Utilizations versus CPU breakdown intensity in the Polling case.
Fig. 4. Mean response times versus CPU breakdown intensity in the Polling case.

Fig. 5. Utilizations versus CPU breakdown intensity in the PPS case.

Fig. 6. Mean response times versus CPU breakdown intensity in the PPS case.
Fig. 7. Utilizations versus CPU breakdown intensity in the PS case.

Fig. 8. Mean response times versus CPU breakdown intensity in the PS case.

\[ Q_i = \sum_{q=0}^{n-1} \sum_{k=0}^{n-k} \sum_{s=1}^{n-k} \sum_{r=1}^{n-k} \sum_{j_1, j_2, \ldots, j_s} \delta(i, j_r)p(q, i_1, \ldots, i_k; j_1, \ldots, j_s). \]

2.2. Numerical results. The results discussed in this section were introduced in [1], where the authors proved by numerical examples that the utilization of the CPU depends on the service discipline (in the case of homogeneous sources), contrary to the reliable systems (see [7]).

In [1], the numerical results were obtained by particular recurrence relations for solving the involved steady-state equations. The advantage of using MOSEL lies in the formulation of the problem in a very short and compact way.

In the following, we assume that the CPU is reliable; however, the terminals are subject to random breakdowns. In the case of \( n = 4, \alpha = 0.001, \) and \( \beta = 999.0, \) the input parameters of different terminals are collected in Table 1.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \lambda_i )</th>
<th>( \mu_i )</th>
<th>( \gamma_i )</th>
<th>( \tau_i )</th>
<th>( \omega_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3500</td>
<td>0.4000</td>
<td>0.2000</td>
<td>0.3000</td>
<td>3.0</td>
</tr>
<tr>
<td>2</td>
<td>0.3500</td>
<td>0.8500</td>
<td>0.2000</td>
<td>0.3000</td>
<td>90.0</td>
</tr>
<tr>
<td>3</td>
<td>0.3500</td>
<td>0.9000</td>
<td>0.2000</td>
<td>0.3000</td>
<td>15.0</td>
</tr>
<tr>
<td>4</td>
<td>0.3500</td>
<td>0.9000</td>
<td>0.2000</td>
<td>0.3000</td>
<td>190.0</td>
</tr>
</tbody>
</table>
In Table 2, we can see that the MOSEL implementation confirms the results of [1] by producing the same performance measures.

In the following, we assume that the CPU is nonreliable, too. In the case of \( n = 4 \), \( \alpha = 0.2 \), and \( \beta = 0.45 \), the input parameters of different terminals remained the same.

In Table 3, we can see how the CPU breakdown intensity affects the performance measures: \( n_j \), \( U_{CPU} \), and terminal utilizations decreased and \( U_j \) and the mean response times of the jobs increased, as was expected. In Figs. 1–8, different utilizations and mean response times are displayed as a function of CPU breakdown intensity employing the FIFO, Polling, PPS, and PS service disciplines at the CPU. The graphics demonstrate the expected behavior of the system. It should be mentioned, however, that the CPU determines the whole functioning via the ratio \( \alpha/\beta \), since the main factor is the stationary probability \( P_{CPU} \) that the CPU is in the operating state. It can easily be seen that

\[
P_{CPU} = \frac{1/\alpha}{1/\alpha + 1/\beta} = \frac{1}{1 + \alpha/\beta},
\]

which is why it is enough to increase \( \alpha \).

| TABLE 2. Performance measures for reliable CPU. |
|-------------|-------------|-------------|-------------|
| \( n_j \) | FIFO | PS | Polling | PPS |
| 1.283976 | 1.230658 | 1.285014 | 1.137364 |
| 0.754239 | 0.767961 | 0.754007 | 0.791032 |
| 0.663056 | 0.660519 | 0.663111 | 0.655889 |
| 0.268280 | 0.249977 | 0.268550 | 0.211706 |
| 0.292608 | 0.313970 | 0.292608 | 0.345383 |
| 0.276354 | 0.269498 | 0.276542 | 0.268845 |
| 0.294113 | 0.318494 | 0.293307 | 0.360611 |
| 3.785136 | 4.397992 | 3.776645 | 6.074148 |
| 2.909238 | 2.322270 | 2.912437 | 1.566298 |
| 3.475490 | 3.650338 | 3.469850 | 3.563455 |
| 2.860429 | 2.210037 | 2.882578 | 1.288600 |

| TABLE 3. Performance measures for nonreliable CPU. |
|-------------|-------------|-------------|-------------|
| \( n_j \) | FIFO | PS | Polling | PPS |
| 1.856720 | 1.786294 | 1.858487 | 1.651324 |
| 0.492754 | 0.507226 | 0.492405 | 0.534534 |
| 0.790920 | 0.789696 | 0.790953 | 0.787215 |
| 0.319337 | 0.287575 | 0.319992 | 0.215116 |
| 0.350007 | 0.390493 | 0.349545 | 0.457172 |
| 0.329285 | 0.317735 | 0.329723 | 0.315673 |
| 0.351974 | 0.398216 | 0.350296 | 0.487978 |
| 4.417234 | 5.340816 | 4.400069 | 8.502297 |
| 3.626719 | 2.809587 | 3.638546 | 1.767214 |
| 4.143941 | 4.425012 | 4.133077 | 4.426307 |
| 2.634280 | 2.673673 | 3.617979 | 1.401484 |

2.3. Summary of performance issues obtained from numerical examples. The calculations using MOSEL showed again that the utilization of the CPU depends on the service discipline (in the case of homogeneous job generation rates), contrary to the reliable systems (see [7]). The different service disciplines generate different performance measures, but at the given setups the sum of utilizations are very close to each other. The whole system behavior basically depends on the stationary probability \( P_{CPU} \) that the CPU is in the operating state. The main performance measures as a function of CPU breakdown intensity demonstrate the expected effect.

3. Conclusions

The MOSEL software package has been used for modeling nonreliable, nonhomogeneous terminal systems. The advanced features of the tool make it possible to formulate the problem in a short and compact form. By giving counter-examples, it has been shown that, contrary to the reliable systems, the utilization of the CPU depends on the service discipline. Sample numerical examples graphically illustrate the effect of CPU failure rate on the main performance measures of the system.
REFERENCES


Department of Mathematics, University of Debrecen, P.O. Box 12, 4010 Debrecen, Hungary
e-mail: almasi@math.klte.hu
e-mail: jsztrik@math.klte.hu